**Amendments to the drawings,**

*There are no amendments to the Drawings.*

RECEIVED
CENTRAL FAX CENTER

OCT 2 6 2006

## Remarks

Status of application

Claims 1-43 were examined and stand rejected in view of prior art. The claims have been amended to further clarify Applicant's invention. Reexamination and reconsideration are respectfully requested.

The invention

A database system with methodology for automated determination and selection of optimal indexes is described. In one embodiment, for example, in a database system having an optimizer for generating an access plan for processing a given database query, an optimizer-based method of the present invention is described for recommending database indexes to be created for optimizing system performance, the method comprises steps of: capturing a workload representative of database queries employed during system use; based on indexes sought by the optimizer during generation of access plans for the database queries, creating virtual indexes for optimizing system performance during execution of the database queries captured in the workload, wherein each the virtual index comprises an in-memory data structure corresponding to a set of potential physical indexes; computing cost benefits for different combinations of the virtual indexes by re-optimizing the workload multiple times, each time eliminating less-beneficial indexes from consideration; and recommending physical indexes to be created based on virtual indexes that have favorable cost benefits for the captured workload.

Prior art rejections

A. Section 102 rejection: Lenzie

Claims 1-43 are rejected under 35 U.S.C. 102(e) as being anticipated by Lenzie ("Lenzie" US Patent 6,728,720). The Examiner's rejection of claim 1 is representative:

> As per claim 1, Lenzie teaches "a method for recommending database indexes to be created for optimizing system performance, the method comprising:" (see Abstract)
>
> "capturing a workload representative of database queries employed

during system use;" (column 5 lines 17-22, wherein a copy is made of all the SQL queries supplied to the engine over a selected period of time)

"creating virtual indexes for optimizing system performance during execution of the database queries captured in the workload;" (column 5 lines 22-28 and column 8 lines 8-13, wherein an index optimizer creates a set of preferred indexes for a database)

"computing cost benefits for different combinations of the virtual indexes;" (column 9 lines 24-40, wherein the cost of indexes are calculated)

"and recommending physical indexes to be created based on virtual indexes that have favorable cost benefits for the captured workload." (column 12 lines 53-59, wherein an index with the highest cost saving is selected).

For the reasons stated below, Applicant's claimed invention may be distinguished on a variety of grounds.

Prior art solutions, including Lenzie, use each virtual index to represent a single possible physical index -- that is, virtual indexes in prior art systems have a one-to-one relationship to physical indexes. Applicant's Index Consultant invention, in contrast, uses an optimizer-based approach where each virtual index to represent a set or class of potential physical indexes. As discussed below, how virtual indexes are created in Applicant's system and what they look like once created differs substantially from prior art approaches.

Consider the basic prior art approach, such as typified by Lenzie. The prior art index optimization system receives a query and then immediately proceeds to extract or parse the interesting pieces (e.g., the predicates and "order by" constructs) by looking at the text of the query. Lenzie and similar systems have a separate algorithm that creates virtual indexes based on the syntax of the queries and the capabilities of the optimizer when the algorithm was written. Based on this preprocessing, the prior art system builds up different virtual indexes that may or may not be helpful to the optimizer.

In Applicant's Index Consultant, the approach is very different: Applicant's system builds the original set of virtual indexes based on the optimizer's needs. The query is not parsed up front but is instead passed on to the database engine's optimizer (i.e., the query is allowed to be processed in a normal manner by the optimizer). Specifically, the optimizer is "watched" for what indexes it needs during "preoptimization" phase. As the optimizer goes through its normal process of deciding what access plan to build for the query, the optimizer will check for the presence of certain indexes. Applicant's Index Consultant does not examine the syntax of the queries in order to create the initial set of virtual indexes but instead lets the optimizer optimize the workload and "watches" it for what types of indexes it is looking for.

Applicant's Index Consultant uses the optimizer check (i.e., whether a particular index exists) to build up virtual indexes, where each virtual index itself represents an entire set or class of potential physical indexes. This approach provides important advantages. Because the optimizer is only interested in general characteristics of an index, Applicant's invention can construct a virtual index that has a more general definition than ones created by prior art systems. In Applicant's system, a single virtual index also includes the "don't care" cases, thus representing a whole class of potential physical indexes. As such, Applicant's Index Consultant is the only solution that represents a virtual index as a set of potential physical indexes. For example, the virtual index {A "don't care", B "don't care"} represents the following set of 8 physical indexes:

(A ASC, B ASC ), (A DESC, B DESC ), (A ASC, B DESC ), (A DESC, B ASC ),
(B ASC, A ASC ), (B DESC, A DESC ), (B ASC, A DESC ), ( B DESC, A ASC )

Prior art solutions, including Lenzie, create each of these 8 indexes as separate virtual indexes. A "virtual index" in Applicant's system represents a whole set of physical indexes, not just a single physical index. This allows, for example, Applicant's system to use a single virtual index structure to represent dozens of physical indexes, instead of the prior art approach of using a single virtual index to represent a single physical index (i.e., 1:1 relationship). The main advantage of this representation is that Applicant's Index Consultant works with a very compact set of virtual indexes and leaves its choices open until the end of the process. As the optimizer (in a database system using Applicant's approach) proceeds through its normal optimization, Applicant's Index Consultant may become more specific about what the final physical index might look like.

Applicant's optimizer-based approach has other advantages. Any modifications made in the optimizer do not affect the Index Consultant. For example, if the optimizer is modified to support a new join method that may make use of a certain type of index, the Index Consultant does not have to be modified or even know about the new join method. Applicant's Index Consultant will see the need for a certain index when the optimizer is looking for it. It will not know --or care -- what the index itself will be used for. Another example is the recommendation of indexes for materialized views that are not referenced in the query text but can be used by the optimizer to answer part of a query. Applicant's Index Consultant will see that the optimizer is looking for an index for a table V which is in fact a materialized view candidate. No other solutions or tools can handle recommendation of indexes for tables which are not referenced in the original workload.

As another advantage, Applicant's Index Consultant collapses its candidate virtual indexes at the end of the process. Given the general representation of a virtual index, the collapsing is very efficient. Additionally, Applicant's Index Consultant is unique in the way the original space requirements are taken into account by dropping 20% (at a time) of the less-useful virtual indexes. The Index Consultant is the only solution available that "reoptimizes" the workload more than twice to discard indexes made not useful by the dropping of the 20% of the indexes.

Applicant's independent claims 1 and 24 have been amended to bring these distinctions to the forefront. For example, the preamble of claim 1 has been amended to emphasize that the claimed method is an optimizer-based approach, as follows (shown in amended form):

> 1. (Currently amended) In a database system <u>having an optimizer for generating an access plan for processing a given database query</u>, a <u>an optimizer-based</u> method for recommending database indexes to be created for optimizing system performance, [...]

Additionally, claim 1 have been amended to emphasize that the virtual indexes are created based on the processing of the optimizer, and that the virtual indexes themselves correspond to sets of potential physical indexes:

> based on indexes sought by the optimizer during generation of access plans for said database queries, creating virtual indexes for optimizing system performance during execution of the database queries captured in the workload, wherein each said virtual index comprises an in-memory data structure corresponding to a set of potential physical indexes;

The feature of Applicant's Index Consultant that "reoptimizes" the workload multiple times to discard less-beneficial indexes (e.g., by dropping of the 20% of the less-useful indexes) is set forth in the amended claim limitation:

> computing cost benefits for different combinations of the virtual indexes by re-optimizing the workload multiple times, each time eliminating less-beneficial indexes from consideration;

(Applicant's other independent claim, claim 24, has been amended in a similar fashion.) It is respectfully submitted that the amended claims set forth a patentable advance over the art.

All told, Applicant's Index Consultant invention constructs virtual indexes in a unique and advantageous manner. This results from the fact that Applicant's approach involves watching the optimizer, as opposed to the prior art approach of simply looking at the syntax of a query. Unless one adopts an approach that requires that the optimizer be watched, one cannot create a system that creates a virtual index comprising a set of physical indexes in the manner done by Applicant's system (and required by Applicant's amended claims). In other words, if the optimizer is not watched, then one cannot discern or know the set of indexes that are available for the optimizer to use. Therefore, at the core architectural level, Applicant's optimizer-based approach is very different from Lenzie and other prior art solutions. As Lenzie teaches an approach that occurs entirely outside of (i.e., irrespective of) the optimizer, the Lenzie teaches, if anything, away from Applicant's claimed approach. In view of the above-discussed amendments and clarifying remarks, it is respectfully submitted that the rejection under Section 102 is overcome.

Any dependent claims not explicitly discussed are believed to be allowable by virtue of dependency from Applicant's independent claims, as discussed in detail above.

Conclusion

In view of the foregoing remarks and the amendment to the claims, it is believed that all claims are now in condition for allowance. Hence, it is respectfully requested that the application be passed to issue at an early date.

If for any reason the Examiner feels that a telephone conference would in any way expedite prosecution of the subject application, the Examiner is invited to telephone the undersigned at 408 884 1507.

Respectfully submitted,

Date: October 26, 2006

[illegible]

Digitally signed by John A. Smart
Date: 2006.10.26 14:21:39 -07'00'

John A. Smart; Reg. No. 34,929
Attorney of Record

408 884 1507
815 572 8299 FAX